Reagan Bennos, Dalton Chenoweth, Stephen Desilets, Cesar Martinez

8/11/2023

**Methodology for Brewery Distribution Problem**

**1. Problem Formulation**

The brewery distribution problem is a transportation optimization problem and formulated as an integer linear program (“ILP”) aimed at minimizing transportation costs while satisfying customer demands for different commodities at various demand points. The objective is to determine the optimal shipping quantities between four distinct breweries and three distinct packaging facilities and between packaging facilities and fifteen different demand points. In an ILP, the decision variables are constrained to take only integer values, which means the variables cannot have fractional or continuous values. In the context of the beer distribution problem, consideration was made for the fact that you can't ship a fraction of a unit of beer; you can only ship whole units.

The decision variables, objective function, and constraints are all defined in sections 2-4 below.

**2. Decision Variables:**

inputs\_1[i][j]: Represents the amount of beer shipped from brewery 'i' to packaging facility 'j'.

inputs\_2[j][k]: Represents the amount of beer shipped from packaging facility 'j' to demand point 'k'.

**3. Objective Function:**

The objective is to minimize the total transportation costs. The objective function aggregates the costs of shipping from breweries to packaging facilities and from packaging facilities to demand points. The cost calculation involves multiplying the quantities of beer shipped by the corresponding shipping costs. The mathematical formulation of the objective function is as follows:

Minimize: Total Transportation Costs

= ∑(i ∈ brewing) ∑(j ∈ packaging) inputs\_1[i][j] × brew\_to\_pack\_shipping\_costs[i][j]

+ ∑(j ∈ packaging) ∑(k ∈ demand\_points) inputs\_2[j][k] × pack\_to\_demand\_shipping\_costs[container][k][j]

**4. Constraints:**

*Demand Constraints*: Ensure that the amount of beer shipped from packaging facilities to demand points meets or exceeds the demand for each commodity at each demand point. The mathematical formulation of the demand constraint is as follows:

∑(j ∈ packaging) inputs\_2[j][k] ≥ demand[c][k], ∀(c ∈ commodities), ∀(k ∈ total\_dp\_demand.keys())

*Brewery Capacity Constraints*: Enforce that the total beer shipped from each brewery to all packaging facilities does not exceed the maximum capacity of the brewery. The mathematical formulation of the brewery capacity constraint is as follows:

∑(j ∈ packaging) inputs\_1[i][j] ≤ brewing\_maximum[i], ∀(i ∈ brewing)

*Packaging Facility Capacity Constraints*: Ensure that the total beer shipped from each packaging facility to demand points does not exceed its maximum capacity. The mathematical formulation of the packaging facility capacity constraint is as follows

∑(k ∈ demand\_points) inputs\_2[j][k] ≤ packaging\_maximum[j], ∀(j ∈ packaging)

*Brewery Minimum Capacity Constraints*: Guarantee that each brewery must produce at least a minimum amount of beer. The mathematical formulation of the brewery minimum capacity constraint is as follows:

∑(j ∈ packaging) inputs\_1[i][j] ≥ brewing\_minimum[i], ∀(i ∈ brewing)

*Packaging Facility Minimum Capacity Constraints:* Guarantee that each packaging facility must ship at least a minimum amount of beer. The mathematical formulation of the packaging facility minimum capacity constraint is as follows:

∑(k ∈ demand\_points) inputs\_2[j][k] ≥ packaging\_minimum[j], ∀(j ∈ packaging)

*Packaging Output Constraints*: Ensure that the total beer shipped from each packaging facility equals the total beer produced by the corresponding breweries. The mathematical formulation of the packaging output constraint is as follows:

∑(k ∈ demand\_points) inputs\_2[j][k] = ∑(i ∈ brewing) inputs\_1[i][j], ∀(j ∈ packaging)

*Liquid Constraints*: Restrict the amount of certain commodities produced based on the availability of specific liquids at the breweries. The mathematical formulation of the liquid constraint is as follows:

∑(j ∈ packaging) inputs\_1[i][j] × commodities[c]["Container"] ≤ liquid\_constraints[required\_liquid]["Max Qty"],

if breweries\_liquids\_config[i][required\_liquid] == 1

*Container Capacity Constraints*: For each container type associated with a commodity, there is a constraint ensuring that the total amount of beer shipped using that container does not exceed the container's maximum capacity. The mathematical formulation of the container constraint is as follows:

∑(k ∈ demand\_points) inputs\_2[j][k] × commodities[c]["Container"] ≤ container\_capacity[required\_container]["Max Qty"],

∀(j ∈ packaging), ∀(c ∈ commodities)

**5. Intended Output**

The team has utilized python code, and specifically the PuLP library, to solve the integer linear programming problem and determine the optimal solution that minimizes transportation costs while satisfying all constraints. The results are printed to the console, including the status of the optimization (e.g., optimal, infeasible), the values of the decision variables, and aggregated brewing and packaging outputs.

**Appendix: Variables and Dictionaries**

*Variables*

inputs\_1[i][j]: The amount of beer shipped from brewery 'i' to packaging facility 'j'.

inputs\_2[j][k]: The amount of beer shipped from packaging facility 'j' to demand point 'k'.

*Dictionaries*

brewing\_maximum: A dictionary containing the maximum beer that can be manufactured for each brewery.

brewing\_minimum: A dictionary containing the minimum units of beer that each brewery must produce.

packaging\_maximum: A dictionary specifying the maximum capacity of each packaging facility.

packaging\_minimum: A dictionary specifying the minimum production capacity of each packaging facility.

total\_dp\_demand: A dictionary containing the aggregate demand for each demand point.

brew\_to\_pack\_shipping\_costs: A list of lists representing shipping costs from breweries to packaging facilities.

pack\_to\_demand\_shipping\_costs: A nested dictionary representing shipping costs from packaging facilities to demand points for different containers.

liquid\_constraints: A dictionary specifying maximum liquid quantity constraints.

breweries\_liquids\_config: A dictionary indicating which breweries use which liquids.

commodities: A dictionary linking commodities to their corresponding liquids and containers.

demand: A dictionary representing demand for each commodity at each demand point.

*Constants*

brewing: A list containing the names of breweries.

packaging: A list containing the labels of packaging facilities.

demand\_points: A list containing the labels of demand points.

*Derived Values*

total\_demand: The total sum of all demand values across demand points.

total\_shipping\_costs: The total transportation costs obtained from the optimized solution.

*Variables and Dictionaries for Optimization*

prob: The PuLP LP problem object representing the beer distribution optimization problem.

routes\_1: A list of tuples representing possible routes from breweries to packaging facilities.

routes\_2: A list of tuples representing possible routes from packaging facilities to demand points.

*Aggregated Results*

brewing\_outputs: A dictionary containing the aggregated amount of beer produced by each brewery.

packaging\_inputs: A dictionary containing the aggregated amount of beer received by each packaging facility.

packaging\_outputs: A dictionary containing the aggregated amount of beer shipped from each packaging facility.